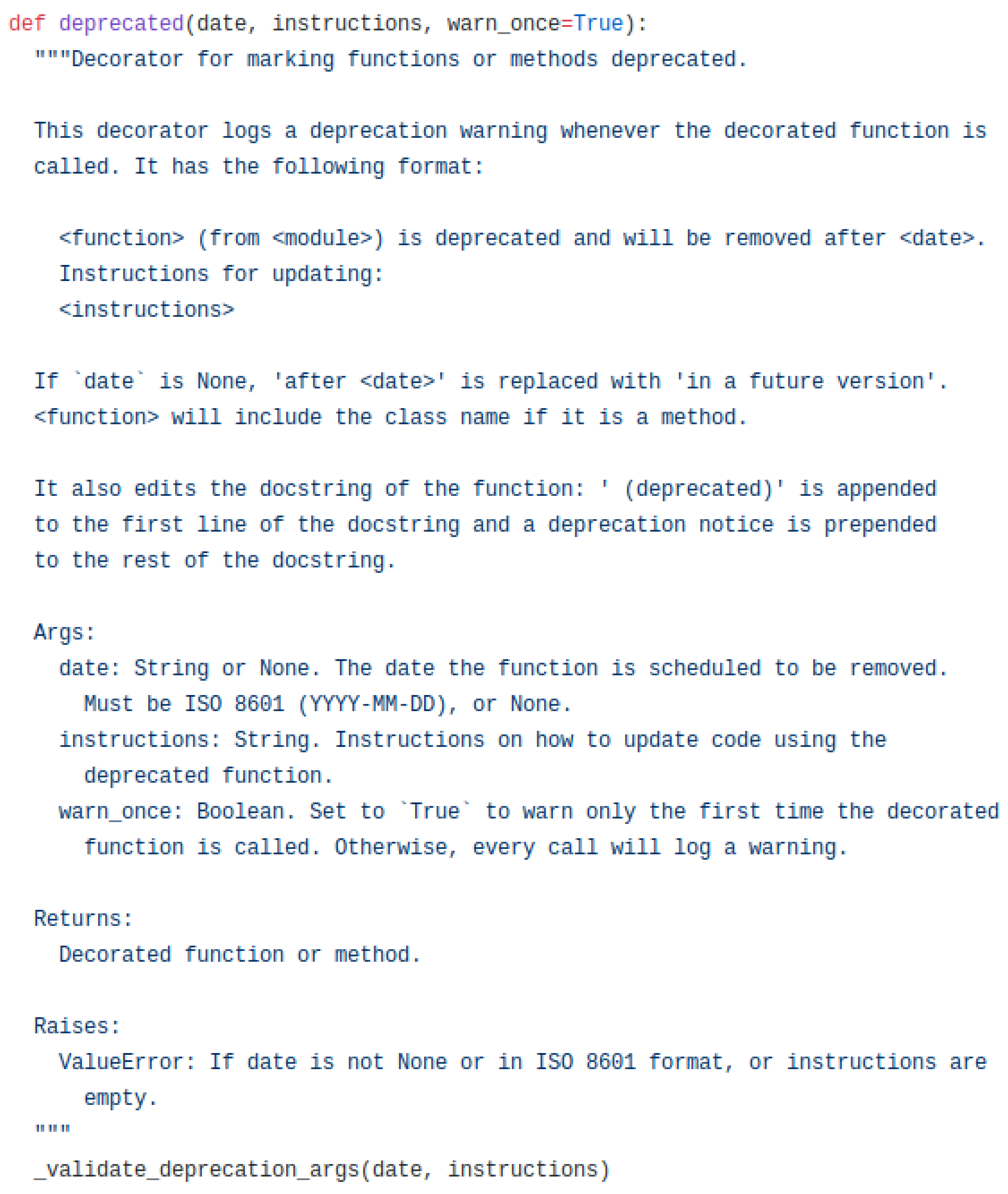
11/4 Design Patterns Activity 2

Team 6

1. Decorator

- Definition: line 274 – 329 in deprecation.py
https://github.com/tensorflow/tensorflow/blob/2917ad1d24cc39a228eac5248ce5d56aafe73f2d/tensorflow/python/util/deprecation.py



```
def deprecated(date, instructions, warn_once=True):
  """Decorator for marking functions or methods deprecated.

  This decorator logs a deprecation warning whenever the decorated function is
  called. It has the following format:

    <function> (from <module>) is deprecated and will be removed after <date>.
    Instructions for updating:
    <instructions>

  If `date` is None, 'after <date>' is replaced with 'in a future version'.
  <function> will include the class name if it is a method.

  It also edits the docstring of the function: ' (deprecated)' is appended
  to the first line of the docstring and a deprecation notice is prepended
  to the rest of the docstring.

  Args:
    date: String or None. The date the function is scheduled to be removed.
      Must be ISO 8601 (YYYY-MM-DD), or None.
    instructions: String. Instructions on how to update code using the
      deprecated function.
    warn_once: Boolean. Set to `True` to warn only the first time the decorated
      function is called. Otherwise, every call will log a warning.

  Returns:
    Decorated function or method.

  Raises:
    ValueError: If date is not None or in ISO 8601 format, or instructions are
      empty.
  """
  _validate_deprecation_args(date, instructions)
```

@deprecated decorator 를 통해 deprecated 된 함수를 호출할 때 deprecated 되었다는 로그 메시지를 출력하는 기능을 추가할 수 있게 해줍니다. 이렇게 함으로써 로그 메시지를 출력하는 코드를 대상 함수의 변경 없이 추가할 수 있습니다.

```
def deprecated_wrapper(func):
  """Deprecation wrapper."""
  decorator_utils.validate_callable(func, 'deprecated')
  @functools.wraps(func)
  def new_func(*args, **kwargs):  # pylint: disable=missing-docstring
    if _PRINT_DEPRECATION_WARNINGS:
      if func not in _PRINTED_WARNING:
        if warn_once:
          _PRINTED_WARNING[func] = True
        logging.warning(
            'From %s: %s (from %s) is deprecated and will be removed %s.\n'
            'Instructions for updating:\n%s',
            _call_location(), decorator_utils.get_qualified_name(func),
            func.__module__,
            'in a future version' if date is None else ('after %s' % date),
            instructions)
    return func(*args, **kwargs)
  return tf_decorator.make_decorator(
      func, new_func, 'deprecated',
      _add_deprecated_function_notice_to_docstring(func.__doc__, date,
                                                   instructions))
return deprecated_wrapper
```

- Usage: line 66 – 73 in logging_ops.py
https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/ops/logging_ops.py

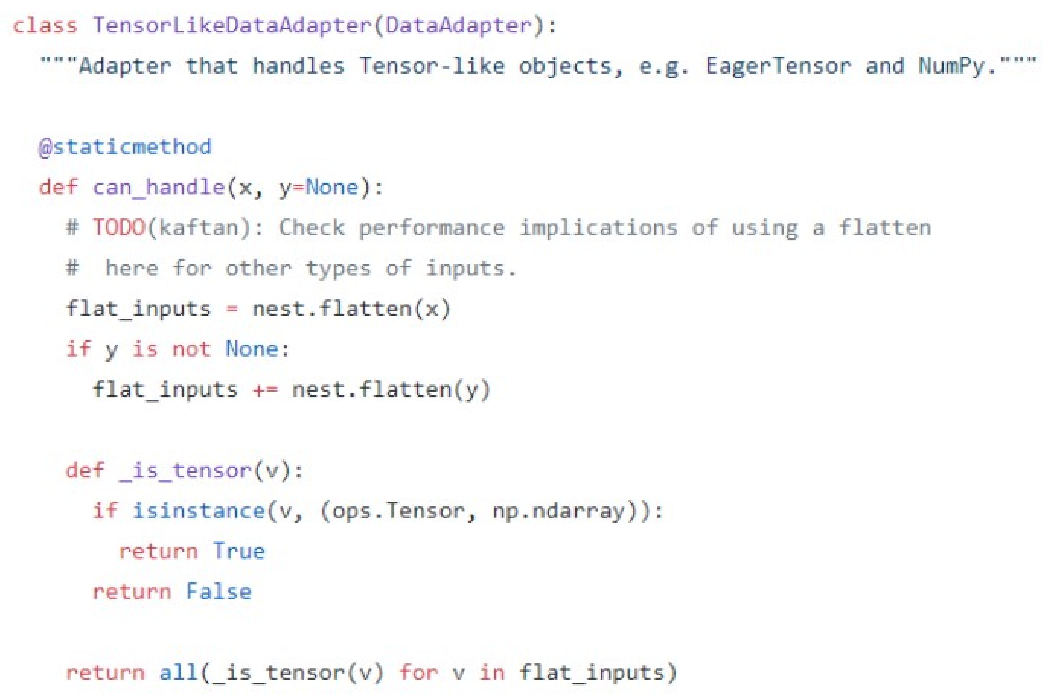
```
@deprecated("2018-08-20", "Use tf.print instead of tf.Print. Note that "
            "tf.print returns a no-output operator that directly "
            "prints the output. Outside of defuns or eager mode, "
            "this operator will not be executed unless it is "
            "directly specified in session.run or used as a "
            "control dependency for other operators. This is "
            "only a concern in graph mode. Below is an example "
            "of how to ensure tf.print executes in graph mode:\n")
@tf_export(v1=["Print"])
def Print(input_, data, message=None, first_n=None, summarize=None, name=None):
  """Prints a list of tensors.
```

이 usage 에선 tf.Print 함수를 호출하면 deprecated 로그 메시지를 출력하여, 대신 tf.print 를 함수를 사용하기를 제안하고 있습니다. @deprecated 에 date 와 instruction 을 parameter 로 전달하면 간편하게 decorator 를 적용할 수 있습니다.

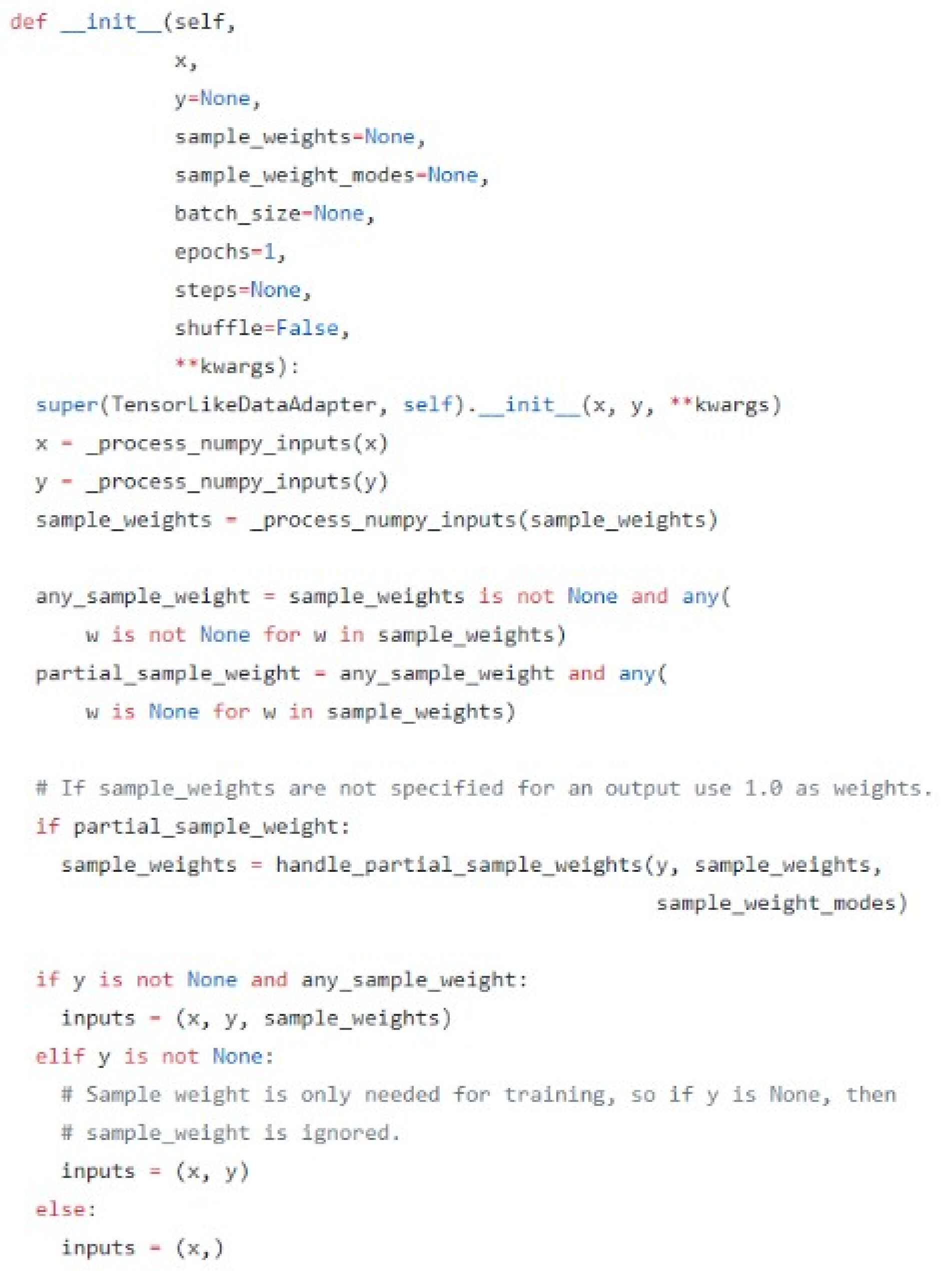
2. **Adapter**

- Definition: in data_adapter.py

https://github.com/tensorflow/tensorflow/blob/83569dfad98f7137dcf38593722fd16674ae632e/tensorflow/python/keras/engine/data_adapter.py



```python
class TensorLikeDataAdapter(DataAdapter):
  """Adapter that handles Tensor-like objects, e.g. EagerTensor and NumPy."""

  @staticmethod
  def can_handle(x, y=None):
    # TODO(kaftan): Check performance implications of using a flatten
    #  here for other types of inputs.
    flat_inputs = nest.flatten(x)
    if y is not None:
      flat_inputs += nest.flatten(y)

    def _is_tensor(v):
      if isinstance(v, (ops.Tensor, np.ndarray)):
        return True
      return False

    return all(_is_tensor(v) for v in flat_inputs)
```

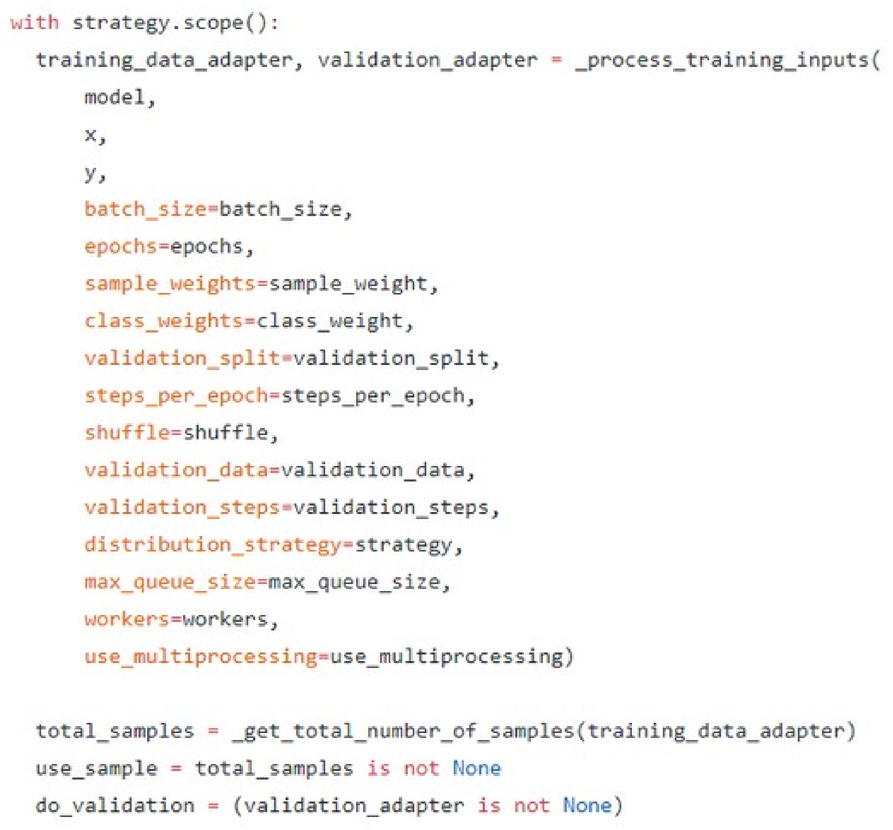


```
def __init__(self,
             x,
             y=None,
             sample_weights=None,
             sample_weight_modes=None,
             batch_size=None,
             epochs=1,
             steps=None,
             shuffle=False,
             **kwargs):
  super(TensorLikeDataAdapter, self).__init__(x, y, **kwargs)
  x = _process_numpy_inputs(x)
  y = _process_numpy_inputs(y)
  sample_weights = _process_numpy_inputs(sample_weights)

  any_sample_weight = sample_weights is not None and any(
      w is not None for w in sample_weights)
  partial_sample_weight = any_sample_weight and any(
      w is None for w in sample_weights)

  # If sample_weights are not specified for an output use 1.0 as weights.
  if partial_sample_weight:
    sample_weights = handle_partial_sample_weights(y, sample_weights,
                                                   sample_weight_modes)

  if y is not None and any_sample_weight:
    inputs = (x, y, sample_weights)
  elif y is not None:
    # Sample weight is only needed for training, so if y is None, then
    # sample_weight is ignored.
    inputs = (x, y)
  else:
    inputs = (x,)
```

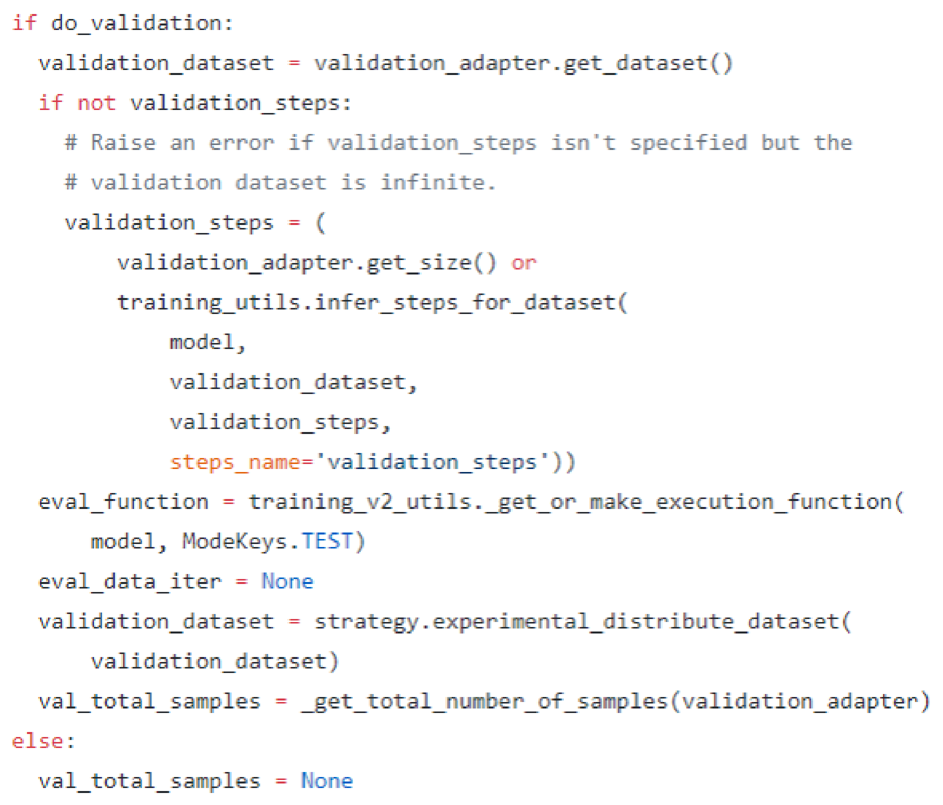
tensorflow 에서는 학습을 하기 위해 Dataset 객체를 만들어 train, validation set 을 만들고 이를 학습에 사용합니다. 사용자가 학습에 사용하는 데이터는 여러 객체(python list, numpy array…)가 있고 이를 모두 다루기 위해 TensorLIkeDataAdapter 를 만들어 사용합니다. DataAdapter class 는 abstract class 로 사용되며 TensorLikeDataAdapter class 에서는 데이터를 다루기 위한 함수를 제공합니다.

init 함수에서는 데이터를 받아 size 등 데이터에 관련된 변수를 설정합니다.

- Usage: line 43 – 58, 617-641 in training_v2.py
https://github.com/tensorflow/tensorflow/blob/a7d5bcf8a1948c90048268770a8b2b643fc0b901/tensorflow/python/keras/engine/training_v2.py



```
with strategy.scope():
  training_data_adapter, validation_adapter = _process_training_inputs(
      model,
      x,
      y,
      batch_size=batch_size,
      epochs=epochs,
      sample_weights=sample_weight,
      class_weights=class_weight,
      validation_split=validation_split,
      steps_per_epoch=steps_per_epoch,
      shuffle=shuffle,
      validation_data=validation_data,
      validation_steps=validation_steps,
      distribution_strategy=strategy,
      max_queue_size=max_queue_size,
      workers=workers,
      use_multiprocessing=use_multiprocessing)

  total_samples = _get_total_number_of_samples(training_data_adapter)
  use_sample = total_samples is not None
  do_validation = (validation_adapter is not None)
```



```
if do_validation:
  validation_dataset = validation_adapter.get_dataset()
  if not validation_steps:
    # Raise an error if validation_steps isn't specified but the
    # validation dataset is infinite.
    validation_steps = (
        validation_adapter.get_size() or
        training_utils.infer_steps_for_dataset(
            model,
            validation_dataset,
            validation_steps,
            steps_name='validation_steps'))
  eval_function = training_v2_utils._get_or_make_execution_function(
      model, ModeKeys.TEST)
  eval_data_iter = None
  validation_dataset = strategy.experimental_distribute_dataset(
      validation_dataset)
  val_total_samples = _get_total_number_of_samples(validation_adapter)
else:
  val_total_samples = None
```

위에서 정의한 data_adapter 를 이용하여 실제 input 을 processing 하여 training 과 validation 을 위한 데이터셋을 구성하기 위해 data_adapter 를 사용하여 디자인한 코드입니다. 아래는 data_adapter 를 사용하기 위해 선언한 global variable 과 process_input 함수의 정의 부분입니다.

```python
# The list of DataAdapter that support validation_split, only numpy and data
# tensor support validation_split for now.
_ADAPTER_FOR_VALIDATION_SPLIT = [data_adapter.TensorLikeDataAdapter,
                                 data_adapter.GenericArrayLikeDataAdapter]

# The list of DataAdapter that support model._standardize_user_data. Currently
# keras.sequence/python generator will cause error when calling
# model._standardize_user_data, this should be updated in future cl, eg, the
# dataset/generate/sequence input will be peeked and processed by
# model._standardize_user_data()
_ADAPTER_FOR_STANDARDIZE_USER_DATA = [
    data_adapter.TensorLikeDataAdapter,
    data_adapter.GenericArrayLikeDataAdapter,
    data_adapter.DatasetAdapter,
    data_adapter.CompositeTensorDataAdapter
]
```

```python
def _process_inputs(model,
                    mode,
                    x,
                    y,
                    batch_size=None,
                    epochs=1,
                    sample_weights=None,
                    class_weights=None,
                    shuffle=False,
                    steps=None,
                    distribution_strategy=None,
                    max_queue_size=10,
                    workers=1,
                    use_multiprocessing=False):
  """Process the inputs for fit/eval/predict()."""
  adapter_cls = data_adapter.select_data_adapter(x, y)
  if adapter_cls in _ADAPTER_FOR_STANDARDIZE_USER_DATA:
    x, y, sample_weights = model._standardize_user_data(
        x,
        y,
        sample_weight=sample_weights,
        class_weight=class_weights,
        batch_size=batch_size,
        check_steps=False,
        steps=steps)
```